Homework 3: Reconstruct

Course: CS 221 Spring 2019
Name: Bryan Yaggi

Setup: n-gram Language Models and Uniform-Cost Search

Our algorithm will base segmentation and insertion decisions on the cost of processed text according to a language model. A language model is some function of the processed text that captures its fluency.

A very common language model in NLP is an n-gram sequence model. This is a function that, given n consecutive words, gives a cost based on to the negative log likelihood that the n-th word appears just after the first $n1$. The cost will always be positive, and lower costs indicate better fluency. As a simple example: in a case where $n = 2$ and c is our n-gram cost function, $c(big, fish)$ would be low, but $c(fish, fish)$ would be fairly high.

Furthermore, these costs are additive; for a unigram model $u(n = 1)$, the cost assigned to $[w1, w2, w3, w4]$ is

$$u(w1) + u(w2) + u(w3) + u(w4).$$

For a bigram model $b(n = 2)$, the cost is

$$b(w0, w1) + b(w1, w2) + b(w2, w3) + b(w3, w4)$$

where $w0$ is -BEGIN-, a special token that denotes the beginning of the sentence.

We have estimated u and b based on the statistics of n-grams in text. Note that any words not in the corpus are automatically assigned a high cost, so you do not have to worry about this part.

A note on low-level efficiency and expectations: this assignment was designed considering input sequences of length no greater than roughly 200 (characters, or list items, depending on the task). Of course, it's great if programs tractably manage larger inputs, but it isn't expected that such inputs not lead to inefficiency due to overwhelming state space growth.

Problem 1: Word Segmentation

In word segmentation, you are given as input a string of alphabetical characters ([a-z]) without whitespace, and your goal is to insert spaces into this string such that the result is the most fluent according to the language model.

(a) Consider the following greedy algorithm: Begin at the front of the string. Find the ending position for the next word that minimizes the language model cost. Repeat, beginning at the end of this chosen segment.

Show that this greedy search is suboptimal. In particular, provide an example input string on which the greedy approach would fail to find the lowest-cost segmentation of the input.

In creating this example, you are free to design the n-gram cost function (both the choice of n and the cost of any n-gram sequences) but costs must be positive and lower cost should indicate better fluency. Note that the cost function doesn't need to be explicitly defined. You can just point out the relative cost of different word sequences that are relevant to the example you provide. And your example should be based on a realistic English word sequence don't simply use abstract symbols with designated costs.

One example would be the input string "basketballismyfavoritesport". If "basket" has a lower cost than "basketball", the result might be "basket ball is my favorite sport" instead of "basketball is my favorite sport". The algorithm will not look ahead to notice that "basketball is" has a lower cost than "basket ball is".

(b) coding

Problem 2: Vowel Insertion

Now you are given a sequence of English words with their vowels missing (A, E, I, O, and U; never Y). Your task is to place vowels back into these words in a way that maximizes sentence fluency (i.e., that minimizes sentence cost). For this task, you will use a bigram cost function.

You are also given a mapping `possibleFills` that maps any vowel-free word to a set of possible reconstructions (complete words). For example, `possibleFills('fg')` returns set([‘fugue’, ‘fog’]).

(a) Consider the following greedy-algorithm: from left to right, repeatedly pick the immediate-best vowel insertion for current vowel-free word given the insertion that was chosen for the previous vowel-free word. This algorithm does not take into account future insertions beyond the current word.

Show, as in question 1-a, that this greedy algorithm is suboptimal, by providing a realistic counter-example using English text. Make any assumptions you’d like about possibleFills and the bigram cost function, but bigram costs must remain positive.

An example would be the input string “Ct njys pttng dgs” for “Cate enjoys petting dogs”. “petting” would not likely be chosen as the vowel-completed word for “pttng” without noticing the following word is likely “dogs”.

(b) coding